# Text Analysis and customer recommendation of women e-commerce clothing reviews

Jihene Samet

23/06/2021

Video link

[Recording] (https://drive.google.com/file/d/1ZIw9ay0Hx0-2X3zKEQGGHgPTlu3cWDbE/view?usp=sharing (https://drive.google.com/file/d/1ZIw9ay0Hx0-2X3zKEQGGHgPTlu3cWDbE/view?usp=sharing))

# 1. Data Description

This data set is from **Kaggle**. It contains information about customer reviews for Women's Clothing E-Commerce. Also, this is a **real commercial data.**

The dataset contains **23486 instances and 11 variables**; 6 quantitative variables and 5 qualitative variables. Each row in the dataset represent customer review.

Description of columns:

**Clothing ID**: Integer Categorical variable that refers to the specific piece being reviewed.

**Age**: Positive Integer variable of the reviewers age.

**Title**: String variable for the title of the review.

**Review Text**: String variable for the review body.

**Rating**: Positive Ordinal Integer variable for the product score granted by the customer from 1 Worst, to 5 Best.

**Recommended IND**: Binary variable stating where the customer recommends the product where 1 is recommended, 0 is not recommended.

**Positive Feedback Count**: Positive Integer documenting the number of other customers who found this review positive.

**Division Name**: Categorical name of the product high level division.

**Department Name**: Categorical name of the product department name.

**Class Name**: Categorical name of the product class name.

**Problematic**: How the review can affect customer's recommendation;

*Importing data set*

```
data<- read.csv("Womens Clothing E-Commerce Reviews.csv", header = T, stringsAsFac
tors = FALSE)
#data structure
str(data)
```

```
## 'data.frame':    23486 obs. of  11 variables:
##  $ X                     : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ Clothing.ID           : int  767 1080 1077 1049 847 1080 858 858 1077 1077
...
##  $ Age                   : int  33 34 60 50 47 49 39 39 24 34 ...
##  $ Title                 : chr  "" "" "Some major design flaws" "My favorite b
uy!" ...
##  $ Review.Text           : chr  "Absolutely wonderful - silky and sexy and com
fortable" "Love this dress!  it's sooo pretty.  i happened to find it in a store,
and i'm glad i did bc i never would have"| __truncated__ "I had such high hopes fo
r this dress and really wanted it to work for me. i initially ordered the petite s
mall "| __truncated__ "I love, love, love this jumpsuit. it's fun, flirty, and fab
ulous! every time i wear it, i get nothing but great compliments!" ...
##  $ Rating                : int  4 5 3 5 5 2 5 4 5 5 ...
##  $ Recommended.IND       : int  1 1 0 1 1 0 1 1 1 1 ...
##  $ Positive.Feedback.Count: int  0 4 0 0 6 4 1 4 0 0 ...
##  $ Division.Name         : chr  "Initmates" "General" "General" "General Petit
e" ...
##  $ Department.Name       : chr  "Intimate" "Dresses" "Dresses" "Bottoms" ...
##  $ Class.Name            : chr  "Intimates" "Dresses" "Dresses" "Pants" ...
```

```
# I drop ped the first column because it's useless for our analysis.
data <- data[-1]
```

# Data pre-processing

*Looking for missing data*

```
sum(is.na(data))
```

```
## [1] 0
```

Since is.na function did not find any NA, I'm guessing that the missing values in my data are filled with empty space.

```
sum(data=="")
```

```
## [1] 4697
```

My guess was right. there are 4697 missing values.Therefore, I'm going to fill the missing values with NA, then I'm going to eliminate them with `na.omit` function.

```
data[data == ""] <- NA
data <- na.omit(data)
dim(data)
```

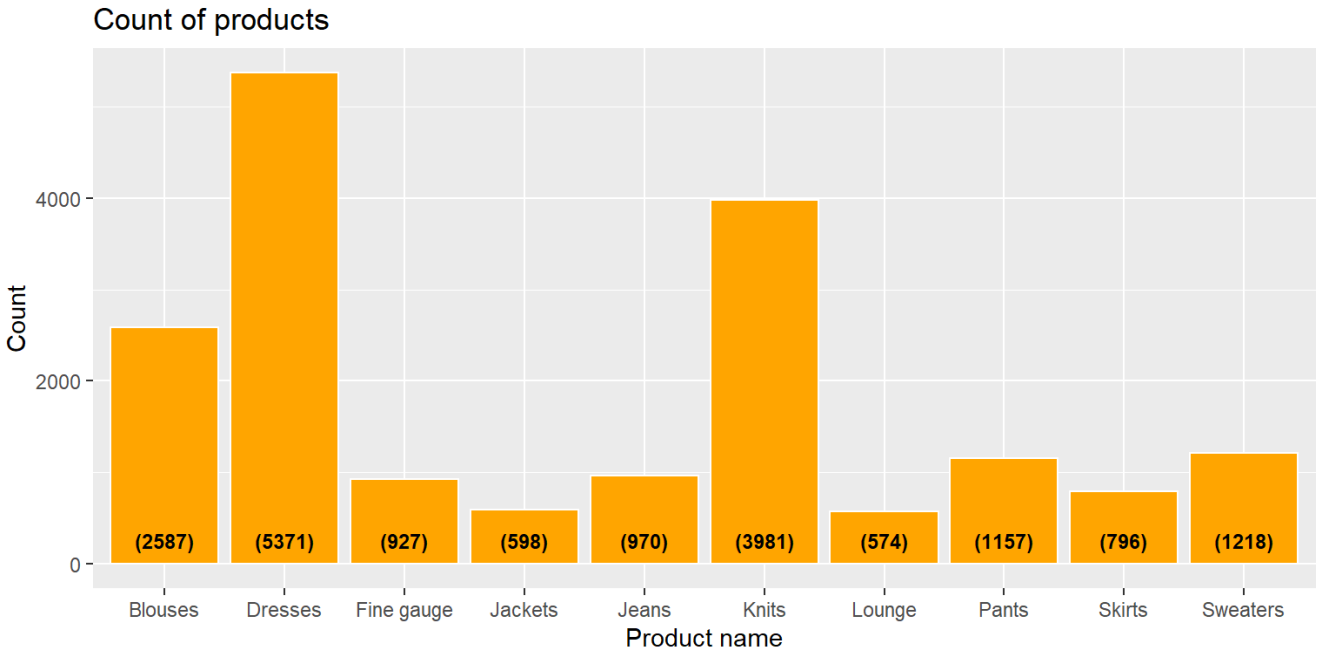
```
## [1] 19662    10
```

**My data is clean to work with. Now, it contains 19662 observations and 10 variables.**

# Data Analysis and Visualization

## Top 10 puchased products

```
library(ggplot2)
library(dplyr) # for pipping
 data %>%
  group_by(Class.Name ) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count)) %>%
   head(10) %>%
  ggplot(aes(x =Class.Name, y = Count )) +
  geom_bar(stat='identity',colour="white" , fill = "orange") +
  geom_text(aes(x = Class.Name, y = 1, label = paste0("(",Count,")",sep="")),
            hjust=0.5, vjust=-1, size = 3, colour = 'black',
            fontface = 'bold') +
  labs(x = 'Product name',
       y = 'Count',
       title = 'Count of products')
```

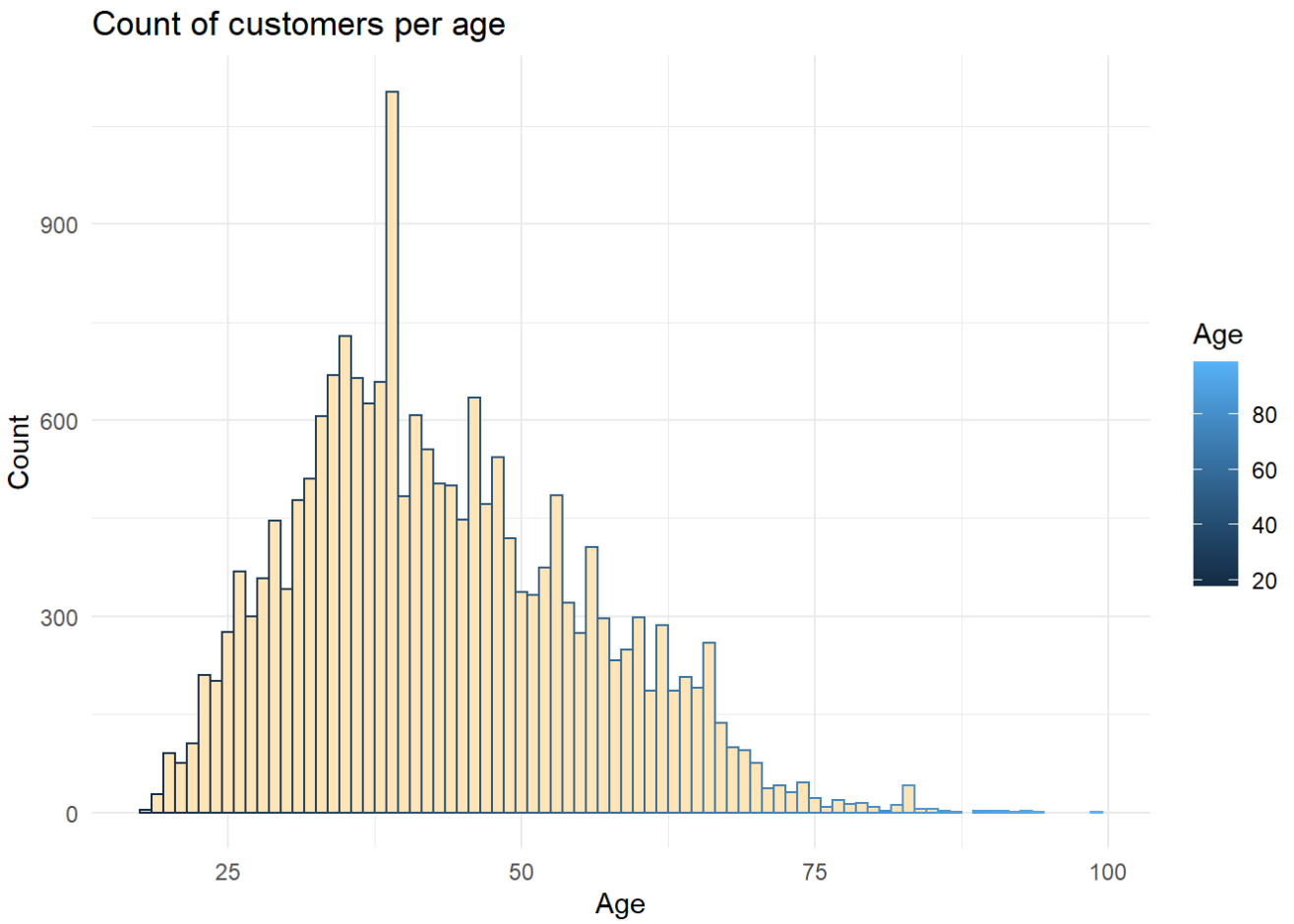



The best seller are Dresses, Blouses and Knits. An important part of the profit comes from these items. This gives gives the company an **idea about the demand**.

## Age distribution

```
data %>%
  group_by(Age) %>%
  summarise(Count = n()) %>%
  ggplot(aes(x = Age, y = Count, color = Age)) +
  geom_histogram(stat = "identity", fill = "moccasin" ,width = 1) +
theme_minimal() +
  labs(title = 'Count of customers per age')
```

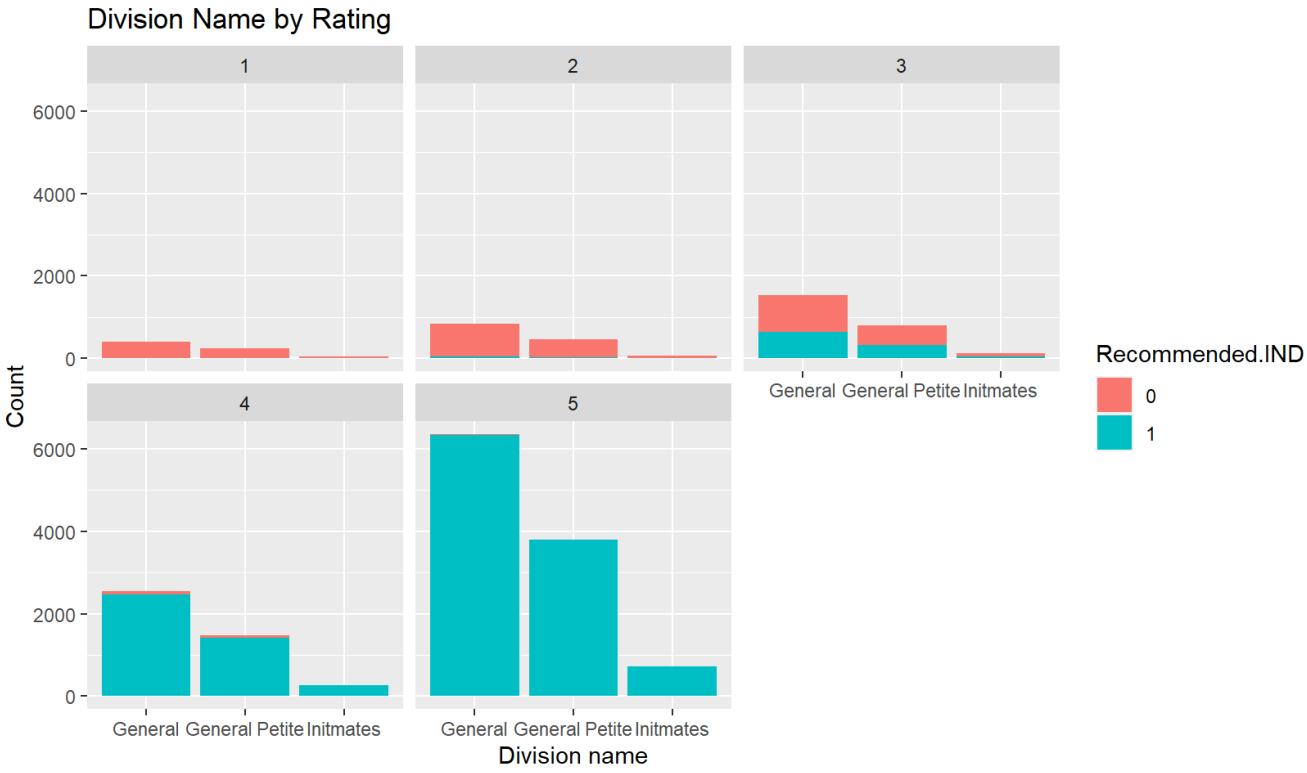
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




People who are in their 30's and 40's are the most frequent buyers, followed by people who are in their 20's and 50's. This gives the company an idea who the **target demographic** is. The younger and the elders are less likely to shop online. It either they are too young to surf the net or too old to adapt with new technology.

## Distribution of Division Name by Rating

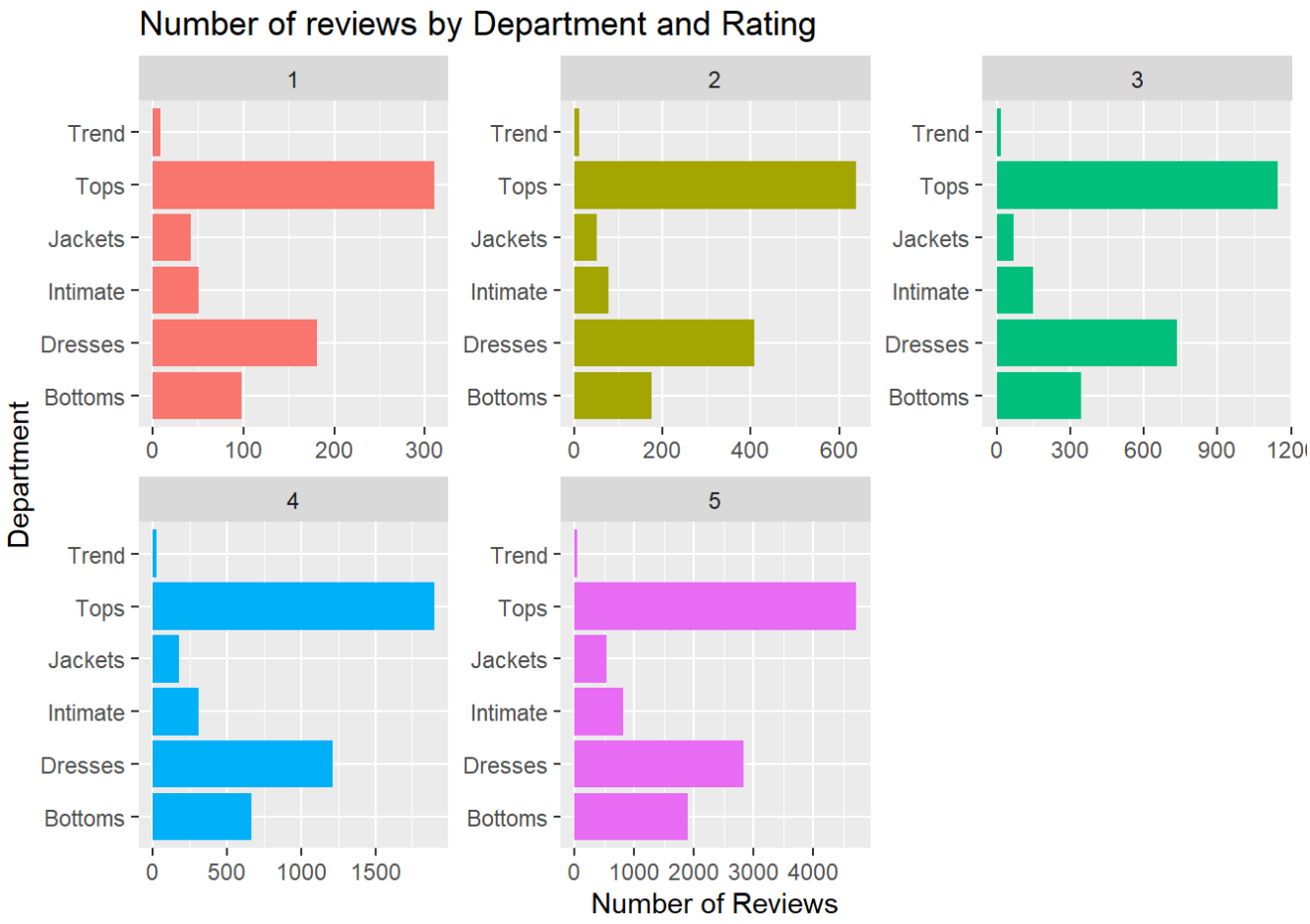
```
data$Recommended.IND <-as.factor(data$Recommended.IND)

ggplot(data) + aes(x= Division.Name, fill=Recommended.IND ) + facet_wrap(~Rating)
+ geom_bar()  +
  labs(x = 'Division name', y = 'Count', title ='Division Name by Rating' )
```




We can see **the chances of recommending the product** from a given division **is higher**, when the product is **rated 5**. Where as a customer who **rated the product three or less** has more chance to **not recommend the product**. Also, the proportion of **5 stars rating is much higher one**. This shows that most of the **customers are satisfied** with the brought item.

## Distribution of number of reviews by Department and Rating

```
#facet wrap by Rating
data$Rating = as.factor(data$Rating)
 data %>% select(Rating, Department.Name) %>%
group_by(Rating) %>% count(Department.Name) %>% ggplot(aes(Department.Name, n, fil
l = Rating)) + geom_bar(stat='identity', show.legend = FALSE) + facet_wrap(~Ratin
g, scales = 'free') +
  labs(x = 'Department', y = 'Number of Reviews', title ='Number of reviews by Dep
artment and Rating' ) + coord_flip()
```




**Tops have the highest number of reviews**, dresses with the second highest. They are maintained within each of the rating group.

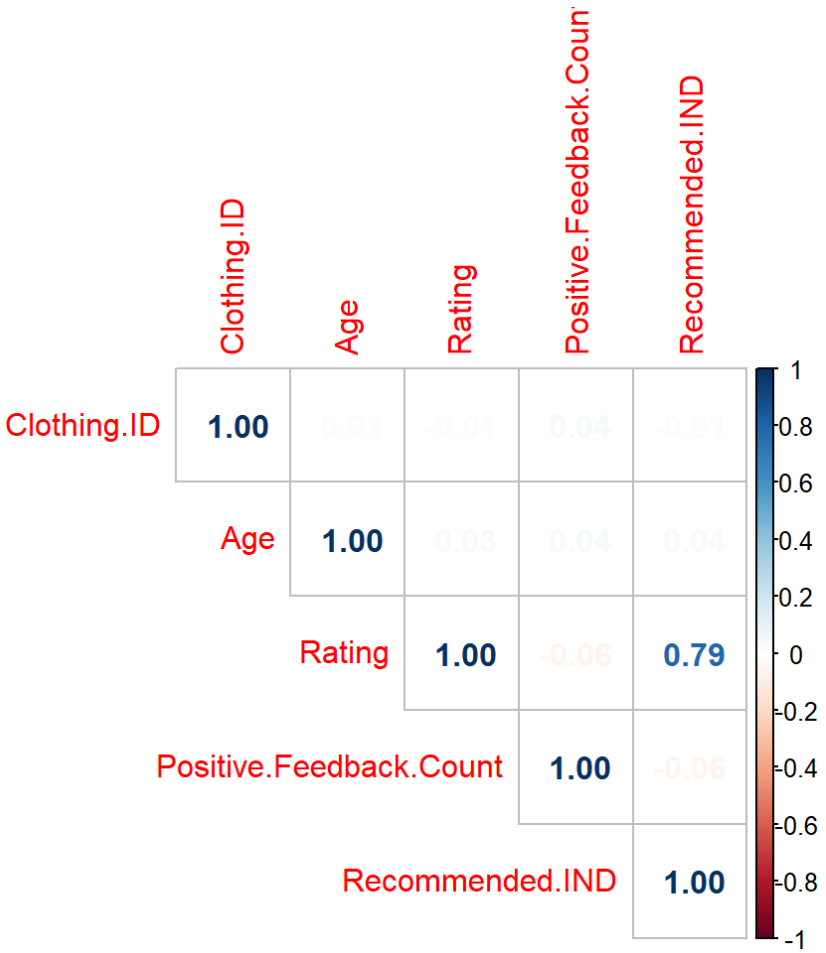
However, the in Trend department 45% rated the product 3 stars or less and 55% rated the product 4 or 5 stars. Which means about **half of the buyers are not satisfied with product from Trend department**.

## Correlation

```
library(corrplot)
```

```
## corrplot 0.89 loaded
```

```
data$Rating = as.integer(data$Rating)
data$Recommended.IND = as.integer(data$Recommended.IND)

corrplot(cor(data[c(1,2,5,7,6)]),
  method = "number",
  type = "upper" # show only upper side
)
```




The correlation between Recommended IND and Rating is 0.79, means that they are highly correlated to each other. In another word, cloths with higher ratings are more recommended.

# Text Extraction and Corpus Creation

## Creating a Corpus

In NLP,in the first we need to convert the text to a Corpus (a collection of text documents). To do that we need to use the `VectorSource` and `Corpus` function from the `tm` library.

```
library(tm) # transforming Words into feature vectors and cleaning
Textcorpus = Corpus(VectorSource(data$Review.Text))
```

1. Text pre-processing

In this step we're going to clean the data in our Corpus. The data clean for text data includes:

. Transforming the data to all lower case letters

. Remove punctuation, numbers, symbols, etc

. Remove stopwords in English literature and remove custom stop words list

. Stemming the document: reducing derived words to their base or root form. for instance, The word "Free", "freedom", and "FREE" should all be treated as the same word "free". This helps a lot in creation of DTM or TDM.

. Remove whitespace.

The transformations are done via the `tm_map()` function.

```
library(SnowballC) #  collapsing words to a common root to aid comparison of vocab
ulary.
Textcorpus [[7]][1]
```

```
## $content
## [1] "I love this dress. i usually get an xs but it runs a little snug in bust s
o i ordered up a size. very flattering and feminine with the usual retailer flair
for style."
```

```
#convert all words in the Corpus to lower case
Textcorpus=tm_map(Textcorpus, tolower)
#remove punctuation
Textcorpus=tm_map(Textcorpus, removePunctuation)
#remove stopwords in english literature
Textcorpus=tm_map(Textcorpus, removeWords, stopwords("english"))
#stem the reviews
Textcorpus=tm_map(Textcorpus, stemDocument)
#Remove context specific stop words
Textcorpus <- tm_map(Textcorpus, removeWords,c("also", "get", "i", "made", "can",
"im", "dress", "just"))
# remove whitespace
Textcorpus=tm_map(Textcorpus, stripWhitespace)
Textcorpus [[7]][1]
```

```
## $content
## [1] "love usual xs run littl snug bust order size flatter feminin usual retail
flair style"
```

The above results show the transformation of the review after cleaning. The punctuation and stopwords are gone and the words are stemmed; *flattering -> flatter*

Now that we have cleaned our review text, we shall proceed to the next step which creating TermDocumentMatrix on the cleaned Corpus..

2. Creating TDM

`TermDocumentMatrix` function creates a matrix in which the rows indicates terms and the columns indicate documents. The data that I'm working on is too big. Therefore, I'll be using `removeSparseTerms` function to reduce the sparsity of the matrix and keep 99.9% of the words that are most frequent.

```
#create term doc matrix
review_tdm <- TermDocumentMatrix(Textcorpus)
review_tdm
```

```
## <<TermDocumentMatrix (terms: 13627, documents: 19662)>>
## Non-/sparse entries: 503185/267430889
## Sparsity           : 100%
## Maximal term length: 32
## Weighting          : term frequency (tf)
```

```
#reduce the sparsity
review_tdm  <- removeSparseTerms(review_tdm , sparse = 0.999)
review_tdm
```

```
## <<TermDocumentMatrix (terms: 1741, documents: 19662)>>
## Non-/sparse entries: 472985/33758557
## Sparsity           : 99%
## Maximal term length: 12
## Weighting          : term frequency (tf)
```

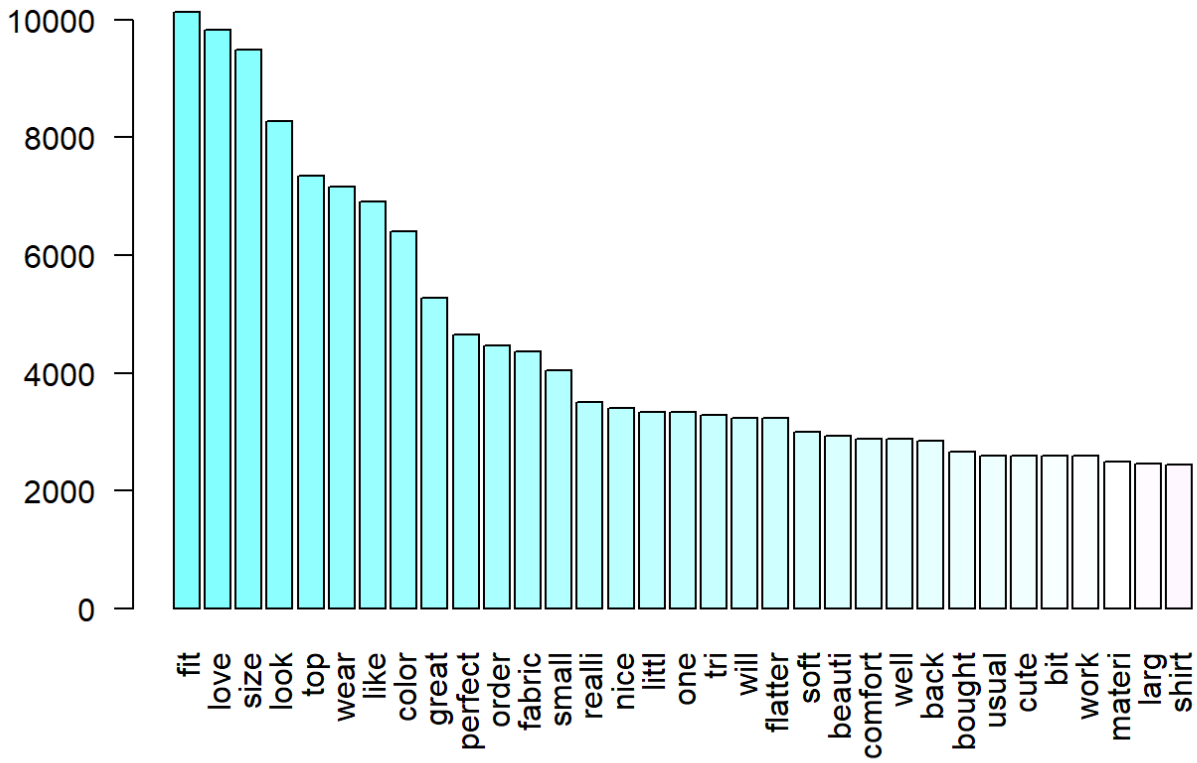
The above results show that the *terms* decreased from 13628 terms to 1741 terms and *Maximal term length* from 32 to 12. Also, Sparcity is no 99%.

The below chunk we will **convert the TDM to a matrix** (term in rows and document in column), then sum all the rows for frequent words. Finally, **sort the frequent words in decreasing order**. Later on we'll see why.

```
# Convert TDM to matrix
review_m <- as.matrix(review_tdm)
# Sum rows frequency data frame
review_term_freq <- rowSums(review_m)
# Sort review_term_freq in descending order
review_term_freq <- sort(review_term_freq, decreasing = T)
```

barchart o Top 20 most common words in review

```
barplot(review_term_freq[1:33],col = cm.colors(61),las = 2)
```




**Love, Fit, size, top** are the **most frequently** used words in the reviews. There were mentioned more that **7000 times**. Followed by wear,color, great, perfect, which are mentioned between 7000 and 4000 times. It indicate that most of the customers are happy.

## WORDCLOUD

Wordclouds can be fitting representation to know the focus of a review(text). The size of words addresses its quantity. The bigger the word the frequent it is. I will be using the `wordcloud` package and `RColorBrewer` package to create a word cloud of the 70 most commonly used words.

```
set.seed(123)
# Create a wordcloud for 'review_term_freq'
library(RColorBrewer)
library(wordcloud)

wordcloud(words= names(review_term_freq),
          freq= review_term_freq,
          max.words= 70,
          random.order=F,
          fixed.asp = T,
          min.freq= 10,
          colors = brewer.pal(9, 'Set1'),
          rot.per= 0.3)
```

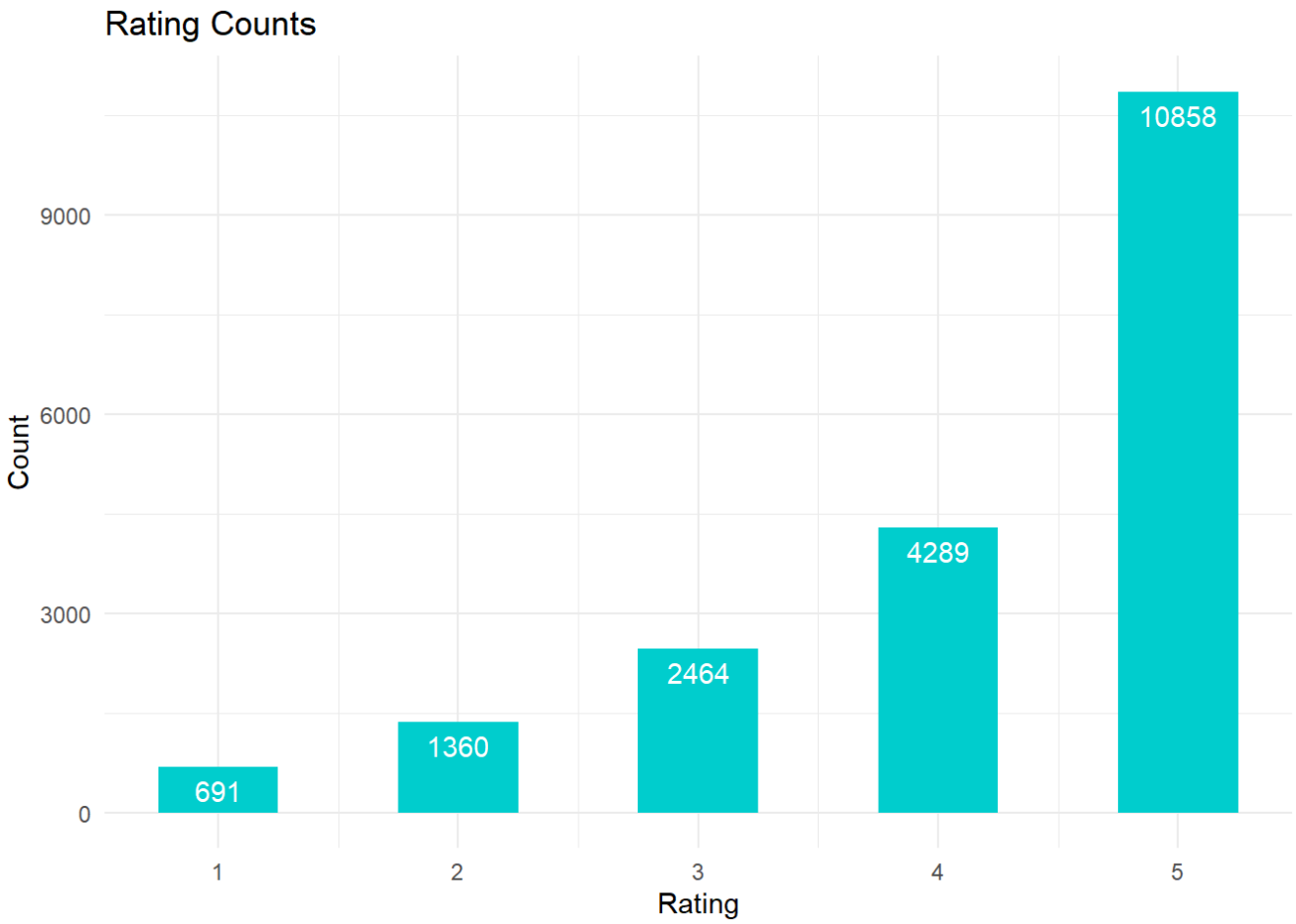

As it shows, there are good(positives) and bad(negative) words.

Things are still a bit blurry now. It's hard to understand the topic with only one word.

The next step we going to analyze wordclouds of the ratings.

## Wordcloud based on Rating

```
# converting Rating variable back to integer
data$Rating <- as.integer(data$Rating)

ggplot(data, aes(x=Rating))+
  geom_bar(stat="bin", bins= 9, fill="cyan3") +
  geom_text(stat='count', aes(label=..count..), vjust=1.6, color="white") +
  ggtitle("Rating Counts") +
  xlab("Rating") + ylab("Count") +
  theme_minimal()
```




5-star reviews are the most popular, and 1-star reviews is wayless common.

We wan to understand more why a customers would rate a product one stars and what he did not like the most. In order for the company to track it weaknesses and improve their products and strategy.

Now we will see the corpus of customers who rated the product 5 stars and those who rated the product one star.

### Word Cloud of 5-star Reviews

In this section we want to determine what are the reason behind rating the product high score or low score. Also, we want to know what did the customers liked the most and what did they dislike if the product was rated one.

#### Subset reviews with 5 starts rating

```
library(tidyverse) # to prepare textual data
library(tidytext)
fiveR<- subset(data$Review.Text , data$Rating==5)
```

We'll be using to same text cleaning as above.

```
# text cleaning
fivecorpus = Corpus(VectorSource(fiveR))
# convert text to lower case
fivecorpus=tm_map(fivecorpus, tolower)
#remove punctuation
fivecorpus=tm_map(fivecorpus, removePunctuation)
# remove ebglish common  stopwords
fivecorpus=tm_map(fivecorpus, removeWords, stopwords("english"))
#stem the reviews
fivecorpus=tm_map(fivecorpus, stemDocument)
# remove specific words
fivecorpus <- tm_map(fivecorpus, removeWords,c("also", "get", "can", "im", "dress"
, "just", "i","like", "company", "made"))
# remove white spaces
fivecorpus=tm_map(fivecorpus, stripWhitespace)
```

## Visualizing Top Trigrams of 5 stars rating reviews

**The following representations of Wordclouds show the most used Tigrams (groups of three consecutive words)** grouped by the different ratings.

```
library(RWeka)
minfreq_trigram <- 5
tritoken <- NGramTokenizer(fivecorpus, Weka_control(min =3, max= 3))
three_word <- data.frame(table(tritoken))
sort_three <- three_word[order(three_word$Freq,decreasing = T),]
wordcloud(sort_three$tritoken,sort_three$Freq,random.order = F,#False; the highest
frequency term will be at the center
          scale=c(3,0.35), #to adjust  word cloud
          min.freq = 10, # minimum frequency
          colors = brewer.pal(10, "Dark2"), max.words = 98) # maximum number of wo
rds  in the wordcloud.
```


Now that we have 3 words, we have a clear view of customers opinions.

The phrases *'fit true size' 'small fit perfect' 'medium fit perfect' ' size fit perfect' and 'fit perfect love'* indicates that all the **size are perfect fit.**

Also, phrases like '*receiv mani compliment' ' compliment evri time and 'got mani compliment'* result in customers are **happy with the product and express the relevance of trend and fashion for customers.**

### Word Cloud of 1-star Reviews

So here basically I'm going to repeat the same steps as above.

```
oneR<- data.frame(txt = data$Review.Text[data$Rating==1], stringsAsFactors = FALSE
)
```

```
onecorpus = Corpus(VectorSource(oneR))
# text cleaning
onecorpus=tm_map(onecorpus, tolower)
onecorpus=tm_map(onecorpus, removePunctuation)
onecorpus=tm_map(onecorpus, removeWords, stopwords("english"))
onecorpus=tm_map(onecorpus, stemDocument)
onecorpus=tm_map(onecorpus, stripWhitespace)
```

Visualizing Top 4-grams of 1 stars rating reviews

```
minfreq_bigram <- 5

bitoken <- NGramTokenizer(onecorpus, Weka_control(min =4, max= 4))
two_word <- data.frame(table(bitoken))
sort_two <- two_word[order(two_word$Freq,decreasing = T),]
wordcloud(sort_two$bitoken,sort_two$Freq,random.order = F, scale=c(1.5,0.3),
          min.freq = 10, colors = brewer.pal(8, "Paired"), max.words = 66)
```


In 1-star ratings, the most common 4-gram were **‘doesnt look anyth like’. ’ hand wash cold water’.**

‘wish read review purchs’ and ‘look noth like model’ express the regret of customers not reading the review before buying and how it looked nothing like what they sow in the site.

to sum up the customers express their disappointment about the quality of the fabric and the difficulties in washing the it.

These reviews can damage the reputation of the product quality, which is a online platforms biggest asset.

# Sentiment analysis

Sentiment analysis is the process of determining the emotional tone behind a phrase. In this context Sentiment analysis is useful to see what people think of the products and to quickly understand consumer attitudes and react accordingly.

In the below chunk we going to obtain sentiment score of 1000 reviews. The score is a calculation of contextual understanding and tone of the text.

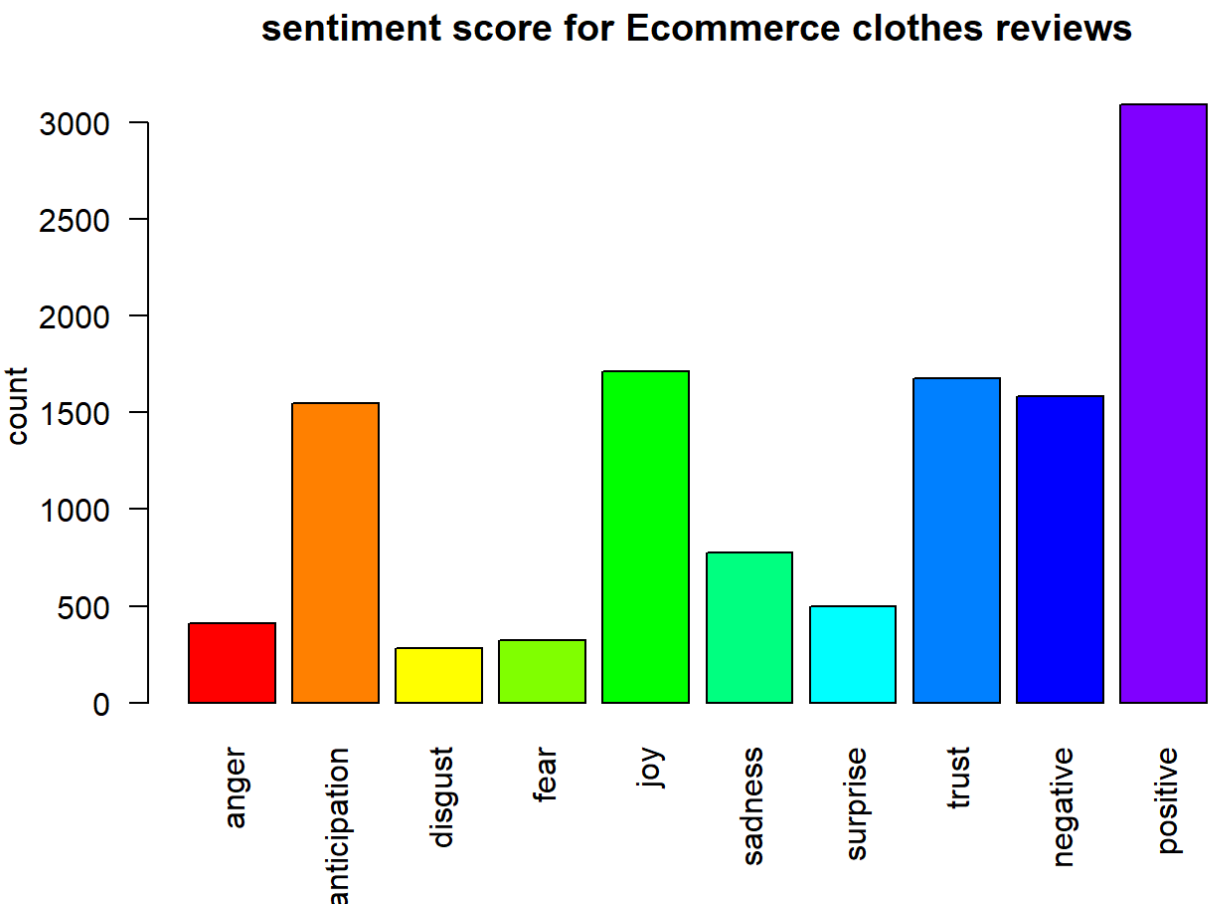
```
library(syuzhet)
library(lubridate)
library(scales)
library(reshape2)
#to obtain sentiment score
s <- get_nrc_sentiment(data$Review.Text[1:1000])
get_nrc_sentiment('love')
```

```
##   anger anticipation disgust fear joy sadness surprise trust negative positive
## 1     0            0       0    0   1       0        0     0        0        1
```

We see that love have a score one for joy. Which result in a score for positive.

Now let's visualize the sentiment score the 1000 selected reviews.

```
barplot(colSums(s),las = 2,col= rainbow(12),ylab = "count", main= "sentiment score for Ecommerce clothes reviews")
```




**The positive reviews dominates the negative ones**? But if we visualize the sentiment of the entire data we will find that the positive reviews dominates the negative one. (I kept it out the analysis due to computational time)

Next, I chose `sentimentr` package developed by Tyler Rinker. In which I used two functions;

`sentiment_by` function to aggregate Sentiment Score for a given text and `highight()` function. Together they gives a html output with parts of sentences highlighted with green and red color to show its polarity.

```
library(sentimentr)
library(tidyverse)

data$Review.Text[1:10] %>%
  get_sentences() %>%
  sentiment_by() %>% #group by score
  highlight()
```

If the negative sentence is more polarize it give a negative score to the review.

# Machine Learning models

This attribute **'Recommended.IND'** is used to classify the customers of dataset in different classes 1 and 0.

**1 stands for recommend the products and 0 not to recommend the product**. The model is used to predict the customer recommendation.

In this analysis, **four** ML models have been taken into consideration; **Naïve Bayes, Decision Tree, Logistic Regression and Random Forest**. Each classification algorithm is applied and the result is compared to the accuracy.

The proposed classification technique is applied to predict customer recommendation based on **Age, Rating, and negative feedback count.**

80 % of the data is used for training of the model and 20% is used for the testing of the model.

First, I made a subset of the needed variables;

```
modeldata <- data[c(2,5,7,6)]
str(modeldata)
```

```
## 'data.frame':    19662 obs. of  4 variables:
##  $ Age                    : int  60 50 47 49 39 39 24 34 53 53 ...
##  $ Rating                 : int  3 5 5 2 5 4 5 5 3 5 ...
##  $ Positive.Feedback.Count: int  0 0 6 4 1 4 0 0 14 2 ...
##  $ Recommended.IND        : int  1 2 2 1 2 2 2 2 1 2 ...
```

```
modeldata$Recommended.IND <-as.factor(modeldata$Recommended.IND)
```

The training set that the ML model is using to train itself and the testing set used to measure the performance of the ML model once built by comparing its predictions to the real values.

```
library(Metrics) # for accuracy
library(caret) # for confusion matrix

#Data Partition
set.seed(1234)
ind <- sample(2, nrow(modeldata), replace = T, prob = c(0.8, 0.2))
train <- modeldata[ind == 1,]
test <- modeldata[ind == 2,]
prop.table(table(train$Recommended.IND))
```

```
##
##         1         2
## 0.182084 0.817916
```

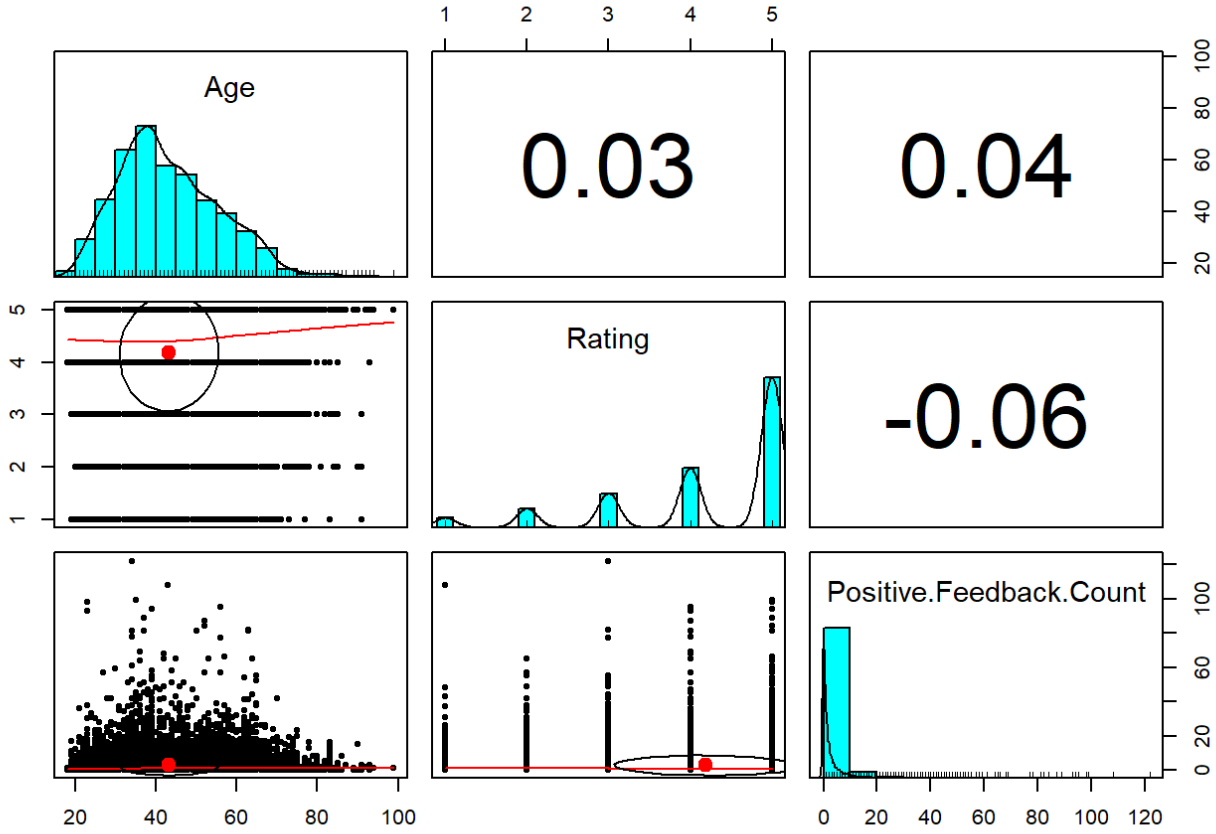
```
prop.table(table(test$Recommended.IND))
```

```
##
##         1         2
## 0.180778 0.819222
```

This appears to be a fairly even split, so we can now build our models.

## Naive Bayes

```
# Libraries
library(naivebayes)
library(psych)
pairs.panels(modeldata[-4])
```




In Bayes model we need to make sure that the independent variables are not highly correlated: its not high we good to go.

```
# Naive Bayes Model
NBmodel <- naive_bayes(Recommended.IND ~ ., data = train)
#predict on train
NBpredict1 <- predict(NBmodel, train)
accuracy(NBpredict1,train$Recommended.IND)
```

```
## [1] 0.9300655
```

```
# Predict on test
NBpredict <- predict(NBmodel, test)
accuracy(NBpredict,test$Recommended.IND)
```

```
## [1] 0.9384694
```

```
# Making the Confusion Matrix
confusionMatrix(table(NBpredict,test$Recommended.IND))
```

```
## Confusion Matrix and Statistics
##
##
## NBpredict    1    2
##         1  683  214
##         2   28 3008
##
##                Accuracy : 0.9385
##                  95% CI : (0.9305, 0.9458)
##     No Information Rate : 0.8192
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8115
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9606
##             Specificity : 0.9336
##          Pos Pred Value : 0.7614
##          Neg Pred Value : 0.9908
##              Prevalence : 0.1808
##          Detection Rate : 0.1737
##    Detection Prevalence : 0.2281
##       Balanced Accuracy : 0.9471
##
##        'Positive' Class : 1
##
```

The results of the accuracy of the train model and test are very close. Hence, we don't have a fitting problem.

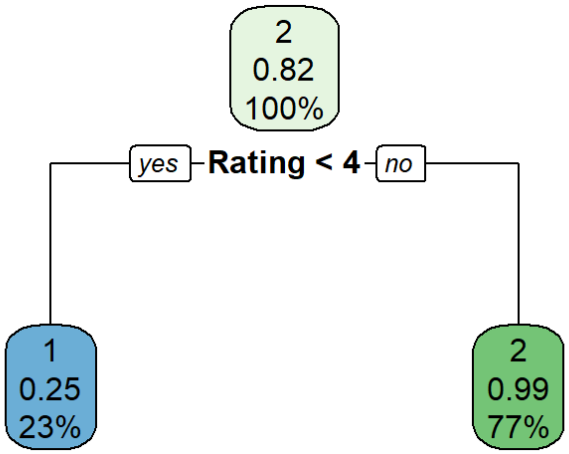
Misclassification is around 6.15%. Testing model accuracy is around 93.85 %

kappa's > 0.75 it's considered excellent

In in model, the classifier correctly predicted 683 0's(not to recommended) and 3008 1's(recommended) the product the test data. However, it incorrectly classified 28 of the 0's(not to recommended) as 1's(recommended).

## Decision Tree

```
set.seed(1234)
library(rpart)
library(rpart.plot)
#Tree Classification
DTmodel <- rpart(Recommended.IND ~., data = train)
rpart.plot(DTmodel)
```




There is only one split in the tree, which is Rating. Rating is the most important variable for classifying customer recommendation into 0( not recommended) and 1 (Recommended).

If the Rating is less than 4, then it's 0( not recommended). For Rating < 4, only 23% were predicted as 0.

It Rating > 4, then it's 1 (Recommended). 77% were predicted as 1.

```
# predict on train
DTpredict1 = predict(DTmodel, newdata=train, type="class")
accuracy(NBpredict,test$Recommended.IND)
```

```
## [1] 0.9384694
```

```
# Predict test
DTpredict = predict(DTmodel, newdata=test, type="class")
accuracy(DTpredict,test$Recommended.IND)
```

```
## [1] 0.9422832
```

```
# Making the Confusion Matrix
confusionMatrix(table(DTpredict,test$Recommended.IND))
```

```
## Confusion Matrix and Statistics
##
##
## DTpredict    1    2
##         1  683  199
##         2   28 3023
##
##                Accuracy : 0.9423
##                  95% CI : (0.9345, 0.9494)
##     No Information Rate : 0.8192
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8218
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9606
##             Specificity : 0.9382
##          Pos Pred Value : 0.7744
##          Neg Pred Value : 0.9908
##              Prevalence : 0.1808
##          Detection Rate : 0.1737
##    Detection Prevalence : 0.2243
##       Balanced Accuracy : 0.9494
##
##        'Positive' Class : 1
##
```

The results of the accuracy of the train model and test are very close. Hence, we don't have a fitting problem.

Misclassification is around 5.77%. Testing model accuracy is around 94.23 %

Decision tree classifier correctly predicted 683 0's(not to recommend) and 3023 1's(recommend) the test data. However, it incorrectly classified 199 of the 0's(not to recommend) as 1's(recommend).

## Logistic Regression

```
#Building the model
set.seed(1234)
glm.fit <- glm(Recommended.IND ~ . , family= binomial, data=train) # Specified bin
omial to indicate logistic regression
# build the  model
LRpredict<- predict(glm.fit, test, type = 'response')

# Evaluating measures of predictive accuracy
tab3 <- table(test$Recommended.IND, LRpredict > .5) # 0.5 is the threshold
tab3
```

```
##
##     FALSE TRUE
##   1   675   36
##   2   195 3027
```

```
LRaccuracy <-sum(diag(tab3))/sum(tab3)
LRaccuracy
```

```
## [1] 0.9412662
```

```
LRSensitivity = tab3[2,2]/sum(tab3[2,])
LRSensitivity
```

```
## [1] 0.9394786
```

```
LRPrecision = tab3[2,2]/sum(tab3[,2])
LRPrecision
```

```
## [1] 0.9882468
```

Misclassification is around 5.88%. Testing model accuracy is around 94.12 % The first row of this matrix considers the response of 0 (not recommended)): 675 were correctly classified as customers who are not recommending the product. While 36 were wrongly classified as 1 (False positive).

# Random Forest

```
set.seed(122)
library(randomForest)
RFclassifier = randomForest(Recommended.IND ~. , data = train, ntree = 55)
# predict train
FRpredict1 <- predict(RFclassifier, train)
accuracy(FRpredict1,train$Recommended.IND)
```

```
## [1] 0.9348337
```

```
# predict test
FRpredict <- predict(RFclassifier, test)
accuracy(FRpredict,test$Recommended.IND)
```

```
## [1] 0.9433003
```

```
confusionMatrix(FRpredict, test$Recommended.IND)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    1    2
##          1  683  195
##          2   28 3027
##
##                Accuracy : 0.9433
##                  95% CI : (0.9356, 0.9503)
##     No Information Rate : 0.8192
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8246
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9606
##             Specificity : 0.9395
##          Pos Pred Value : 0.7779
##          Neg Pred Value : 0.9908
##              Prevalence : 0.1808
##          Detection Rate : 0.1737
##    Detection Prevalence : 0.2232
##       Balanced Accuracy : 0.9500
##
##        'Positive' Class : 1
##
```

The results of the accuracy of the train model and test are very close. Hence, we don't have a fitting problem.

Lastly, I examined **F1-score** for each algorithm.

```
2*(0.9606*0.7614 )/(0.9606+0.7614 )
```

```
## [1] 0.8494783
```

```
2*(0.9606*0.7744 )/(0.9606+0.7744 )
```

```
## [1] 0.8575085
```

```
2*(0.9882*0.9395 )/(0.9882+0.9395 )
```

```
## [1] 0.9632348
```

```
2*(0.9606*0.7779 )/(0.9606+0.7779 )
```

```
## [1] 0.85965
```

## Model selection

The table shows comparison of ML algorithms metrics:

| ML model | Accuracy | Sensitivity | Precision | F1 score |
|---|---|---|---|---|
| Naive Bayes | 0.9385 | 0.9606 | 0.7614 | 0.8495 |
| Decision tree | 0.9423 | 0.9606 | 0.7744 | 0.8575 |
| Logistic Regression | 0.9413 | 0.9882 | 0.9395 | 0.9632 |
| Random Forest | 0.9433 | 0.9606 | 0.7779 | 0.8597 |

The greater the accuracy and F1 Score, the better is the performance of our model.

To minimizing False Negatives,it would be better if Recall is close to 1.

To minimizing False Positives, it would be better if Precision is close to 1.

When we look at the results of the evaluation metrics, **Logistic Regression and Random forest gives the best results for our analysis**. Thus, both of them are very effective at predicting.

The **highest accuracy is 94.33%** of Random Forest classifier and the **lowest accuracy is 93.85% of Naive Bayes classifier.**

**Logistic regression model have the highest Precision, Sensitivity and F1 score.**

Therefore, Logistic regression is selected for classification and prediction in the future data sample.

# Conclusion

- The most brought products are dresses and knits and from General division they also dominate in rating either good rate or bad rate. Which gives an idea about the supply and demand.
- Most important customers are in their 30’s and 40’s. Knowing the demographics of the clients can help in decision making. For instance online advertisements for sites access by these customers.
- Positive reviews are empty of criticism. They are busy with confirming the fit and getting compliments.
- On the other hand, negative reviews express disappointment criticizing the fabric and the washing method(with cold water). These negatives reviews and this may damage the reputation of store.
- Finally Logistic Regression and Random Forest model are the perfect ML model to predict customer recommendation.